π_0-EqM: Equilibrium Matching for Closed-Loop Vision-Language-Action Control

Huanming Liu[1,*], Congsheng Xu[2,*], Jianmin Ji[1,†], and Yao Mu[2,†]

***Abstract*— Currently, Vision-Language-Action (VLA) models have become the most adopted paradigm for robotic manipulation for its great potential for task generalization. While most generative flow-matching action decoders for VLA control are often deployed with fixed sampling horizons, limiting state-dependent compute and temporal reuse across control cycles. We present π_0-EqM, which replaces the flow-matching expert in π_0 with an Equilibrium Matching (EqM) decoder while leaving the upstream VLA stack unchanged. Under a matched 300-step budget, π_0-EqM improves RoboTwin average success from 40.4% to 50.2% across 19 tasks and remains competitive on LIBERO, with its clearest gain on LIBERO-10 (87.0%). Two threshold scans reveal a task-dependent *non-monotonic* relation between residual and success, which we term the *stationarity–executability gap*. The results suggest that inference depth in iterative VLA control is part of policy design and introduce an energy-based VLA perspective that may inform future work on composable action generation across tasks and embodiments.**

I. Introduction

Vision-Language-Action (VLA) policies map visual observations and language instructions directly to actions [1], [2], [3], [4]. In these systems, a central design choice is the action decoder that produces short action chunks for receding-horizon execution. Recent large-scale VLAs such as π_0 use flow-matching decoders [5], [6], while diffusion-based action generation remains a strong alternative [7]. These decoders are expressive, but are typically deployed with fixed denoising or integration horizons.

This is restrictive in closed loop. Different states benefit from different amounts of computation, and consecutive control cycles are strongly correlated, making iterate reuse desirable. Yet in diffusion and flow decoders, intermediate states are tied to explicit time or noise semantics, which makes early stopping and cross-cycle warm starts awkward.

We therefore ask whether inference depth should be treated as part of deployed policy design. To study this, we replace the flow-matching expert in π_0 with an Equilibrium Matching (EqM) decoder [8]. EqM learns a time-free conditional vector field and decodes actions by iterative equilibrium solving rather than time-indexed transport.

This also introduces an energy-based perspective into VLA and points to composable energy-based action generation across tasks and embodiments as a natural extension. Because EqM is time free, intermediate iterates no longer carry explicit diffusion-time semantics, making stopping and reuse deployment decisions rather than sampler-specific interventions.

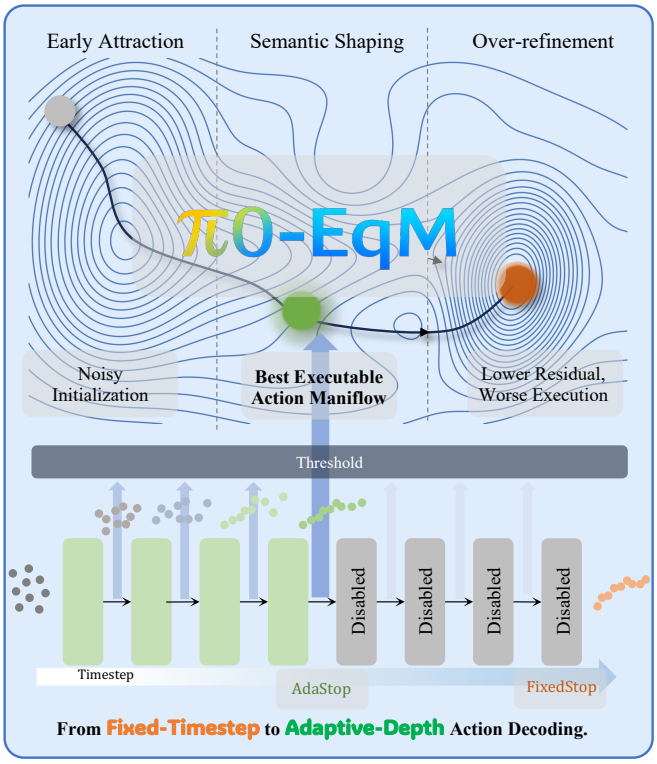

Fig. 1. Overview of π_0-EqM. We replace only the action decoder in π_0 and cast action generation as iterative equilibrium solving, enabling adaptive stopping and warm starts. An executable intermediate action may appear before full numerical convergence.

Under a matched 300-step inference budget, π_0-EqM improves RoboTwin average success from 40.4% to 50.2% across 19 tasks. On LIBERO, it remains competitive overall and improves LIBERO-10 from 85.2% to 87.0%. Threshold scans further show that lower residual does not uniformly imply better execution. We refer to this mismatch as the *stationarity–executability gap*.

Our contributions are summarized as follows:

- We introduce π_0-EqM, a decoder replacement for π_0 that improves RoboTwin average success by 9.8 points under a matched 300-step budget while remaining competitive on LIBERO.
- We show that stopping depth can change closed-loop behavior in EqM-based control.
- We connect a local solver analysis to threshold-scan behavior and use the stationarity–executability gap to interpret iterative action decoding.

[1]Huanming Liu and Jianmin Ji are with University of Science and Technology of China, China lhm_0410@mail.ustc.edu.cn, jianmin@ustc.edu.cn

[2]Congsheng Xu and Yao Mu are with Shanghai Jiao Tong University, China acondaway@sjtu.edu.cn, muyao@sjtu.edu.cn

*Equal contribution.

†Corresponding authors.

- Our method qualitatively and quantitatively shows that the time-independent diffusion action generation method can lead to better performance for robotic manipulation.

II. Related Work

a) VLA policies and chunked action decoding: Modern VLA systems combine visual encoders, language conditioning, and chunked action prediction in a single model interface [1], [2], [3], [4]. Action chunking improves stability in long-horizon control [9] and mitigate the non-marcov property of the robotic manipulation. Usually, VLAs utilizes a web-scale data pre-trained VLM as the understanding backbone and discrete action token output, and a light-weight head for continuous action generation based on the VLM's visual and action semantic information. π series models extends this line with a flow-matching action expert [5]; our work keeps the structure of upstream VLM and modifies only the action decoder that follows the design art of Equilibrium Matching.

b) Optimization-based control and energy-based inference: A complementary line treats action generation as optimization, either by solving a control objective online or by minimizing a learned conditional energy / following its gradient field [10], [11], [12], [13]. Compared with fixed-horizon generative decoding, such formulations make the iterative structure of inference explicit and render test-time computation more controllable, which is particularly relevant in closed-loop settings. EqM is related to this family in its inference form, but differs in parameterization: instead of requiring an explicit scalar energy, it learns an implicit conditional vector field whose equilibria define the target action chunks [8]. This makes EqM a natural fit for exposing adaptive stopping and warm-start reuse within a VLA decoder.

III. Method

A. Preliminaries: Action Decoding as Iterative Inference

At control step t, a VLA policy maps the multi-modal condition $c_t = (o_t, \ell, s_t)$ to an action chunk $A_t = [a_t, a_{t+1}, \ldots, a_{t+H-1}] \in \mathbb{R}^{H\times d}$. In our receding-horizon setup, only the immediate action a_t is executed before replanning.

Rather than treating decoding as a fixed-cost black box, we view it as iterative inference whose depth is a controllable policy variable. EqM provides the mathematical scaffolding for this view.

B. Equilibrium Matching for VLA Control

Unlike diffusion or flow-matching policies that parameterize time-indexed score or velocity fields, EqM learns a **time-invariant** (autonomous) conditional vector field $f_\theta(A; c_t) \in \mathbb{R}^{H\times d}$. The roots of this field correspond to the target equilibrium action chunks.

a) Training Objective: Given a demonstration chunk $A \sim p_{\text{data}}$, structural noise $\varepsilon \sim \mathcal{N}(0, I)$, and an interpolation factor $\gamma \sim \mathcal{U}(0,1)$, we define the interpolant $A_\gamma = \gamma A + (1-\gamma)\varepsilon$. The field f_θ is trained to match a targeted vector $v_{\text{target}}(\gamma; A, \varepsilon) = w(\gamma)(A - \varepsilon)$ via the EqM loss:

$$\mathcal{L}_{\text{EqM}}(\theta) = \mathbb{E}_{\gamma,A,\varepsilon}\left[\left\| f_\theta(A_\gamma; c_t) - w(\gamma)(A-\varepsilon)\right\|_2^2\right], \quad (1)$$

where $w(\gamma)$ is a scheduling scalar designed to decay to zero as $\gamma \to 1$. This enforces the data manifold to act as a set of stable equilibria.

b) Inference as Equilibrium Solving: At test time, actions are recovered by solving $f_\theta(A; c_t) = 0$. Starting from $A_t^{(0)}$ and setting $A_t^{(-1)} = A_t^{(0)}$, we use a Nesterov-accelerated solver:

$$\widetilde{A}_t^{(k)} = A_t^{(k)} + \mu\left(A_t^{(k)} - A_t^{(k-1)}\right), \quad (2)$$

$$A_t^{(k+1)} = \widetilde{A}_t^{(k)} - \eta f_\theta(\widetilde{A}_t^{(k)}; c_t), \quad (3)$$

where $\mu \in [0,1)$ is the momentum factor and $\eta > 0$ is the step size. We use the same lookahead point $\widetilde{A}_t^{(k)}$ for both the update and the residual check.

C. Temporal Coherence: Adaptive Stopping and Warm-Start

The iterative, time-invariant nature of EqM exposes two deployment mechanisms:

Adaptive Inference Depth: We define the normalized lookahead residual

$$r_k = \frac{\left\| f_\theta\left(\widetilde{A}_t^{(k)}; c_t\right)\right\|_2}{\sqrt{Hd}}, \quad (4)$$

and stop at $T_t(\tau) = \min\{k \geq 0 : r_k \leq \tau \text{ or } k = K_{\max}\}$. This exposes state-dependent test-time compute without changing the upstream VLA stack. Using the same lookahead residual in the stopping rule, theory, and threshold scans also keeps the numerical quantity consistent across analysis and deployment.

Structural Warm-Start: Because EqM decoding is independent of a fixed noise schedule, we can exploit temporal coherence between consecutive control cycles. In our implementation, we use a partial warm start: a half-length segment of the previous output chunk is copied into the first half of the current initialization, and the second half is filled with fresh random noise:

$$A_t^{(0)} = \left[A_{t-1,\text{copy}}^{(\text{out})},\ \varepsilon_{\text{tail}}\right], \qquad A_{t-1,\text{copy}}^{(\text{out})}, \varepsilon_{\text{tail}} \in \mathbb{R}^{\frac{H}{2}\times d}, \quad (5)$$

where $A_{t-1}^{(\text{out})}$ is the final action chunk returned at control step $t-1$, $A_{t-1,\text{copy}}^{(\text{out})}$ denotes the reused half-chunk from that prediction, and $\varepsilon_{\text{tail}}$ is sampled random noise for the remaining half of the horizon. This preserves short-term temporal structure while allowing the remaining horizon to be reinitialized under the new observation. In practice, such warm starts can place the initial guess closer to a local equilibrium basin and reduce the iterations needed to reach a target residual.

D. Theoretical Interpretation of Solver Iterates

To connect the solver to the stopping rule used later in Sec. V-C, we provide a local interpretation of early stopping.

Proposition 1 (Local Potential Descent). *Assume that within a neighborhood of the inference trajectory for a fixed c_t, the learned field behaves as the gradient of a surrogate energy, $f_\theta(A;c_t) = \nabla E_t(A)$, where $E_t(A)$ is L-smooth and lower-bounded by $E_t^\star$. For any step size $\eta \le 1/L$, the iterates generated by Eq. (3) (setting $\mu = 0$, so that $\widetilde{A}_t^{(k)} = A_t^{(k)}$) follow a descent path $E_t(A_t^{(k+1)}) \le E_t(A_t^{(k)})$. Using the normalized residual r_k from Eq. (4), after K steps the minimum residual is bounded by:*

$$\min_{0\le j<K} r_j^2 \le \frac{2\left(E_t(A_t^{(0)}) - E_t^\star\right)}{\eta K H d}. \tag{6}$$

Moreover, if the solver map is locally contractive with factor $\rho \in (0,1)$ near an equilibrium $A_t^\star$, then

$$\left\| A_t^{(k)} - A_t^\star \right\|_2 \le \rho^k \left\| A_t^{(0)} - A_t^\star \right\|_2. \tag{7}$$

Under the same local L-smoothness assumption and the identity $f_\theta(A_t^\star;c_t) = 0$, this implies

$$r_k \le \frac{L}{\sqrt{Hd}} \rho^k \left\| A_t^{(0)} - A_t^\star \right\|_2. \tag{8}$$

Therefore, a sufficient number of iterations to reach the experimental stopping threshold $r_k \le \tau$ is

$$K_\tau^{\text{suff}} = \left\lceil \frac{\log\left(\frac{L\left\| A_t^{(0)} - A_t^\star \right\|_2}{\tau\sqrt{Hd}} \right)}{\log(1/\rho)} \right\rceil, \tag{9}$$

whenever the logarithm argument exceeds 1; otherwise the threshold is already met at initialization. Warm-starting reduces this sufficient iteration bound by reducing the initial distance $\left\| A_t^{(0)} - A_t^\star \right\|_2$, up to the ceiling effect in Eq. (9).

This analysis is intentionally local. It justifies the residual threshold used in the experiments as a numerically meaningful stopping rule and explains why warm starts can reduce the iterations needed to reach a target threshold, but it does not determine which threshold maximizes task success. That behavioral question is addressed empirically in Sec. V-C.

Eq. (9) also makes the warm-start effect explicit. Ignoring the ceiling effect, if the initialization distance is reduced by a factor $\alpha \in (0,1)$, then the sufficient iteration count decreases by roughly $\log(1/\alpha)/\log(1/\rho)$. In other words, warm starts are most valuable when the local contraction is strong and the target threshold is strict, which matches the practical intuition that temporal coherence matters most in hard, iterative control regimes.

IV. Experimental Setup

We evaluate on the 19-task RoboTwin benchmark [14]; representative video keyframes are provided in the supplementary material. Each task uses 500 demonstrations (100 clean and 400 randomized), for 9,500 total. We compare the original π_0 against π_0-EqM while keeping the upstream representation stack and conditioning interface fixed, so only the action expert changes. The main comparison uses a matched 300-step inference budget. Threshold-scan experiments allow up to 1000 EqM iterations and stop when the normalized lookahead residual falls below τ.

We also evaluate on the standard LIBERO suites [15]: LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-10. For LIBERO, we report suite-level success rates against the corresponding π_0 baseline.

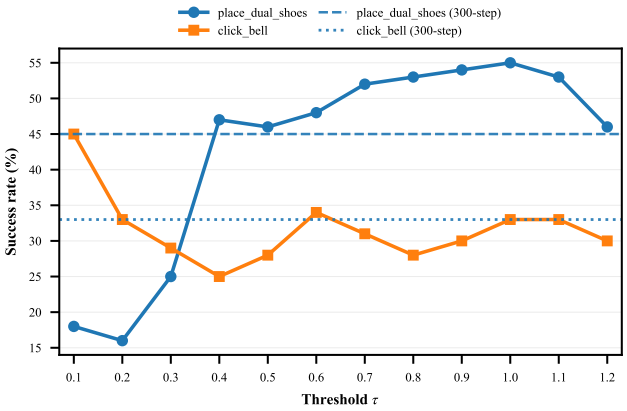

Fig. 2. Threshold scans on two RoboTwin tasks. The preferred threshold differs across tasks, so the same residual threshold changes both solver cost and closed-loop behavior.

V. Results

A. EqM improves the flow-matching baseline on RoboTwin

Table I reports the 19-task deployment results. Under a matched 300-step budget, π_0-EqM improves 12 tasks, degrades 5, and ties 2, raising the average success rate from 40.4 to 50.2. The largest gains occur on `pick_dual_bottles`, `place_cans_plasticbox`, and `put_bottles_dustbin`.

B. LIBERO suite results

Table II reports the LIBERO results. π_0-EqM improves from 96.8% to 97.2% on LIBERO-Spatial and from 85.2% to 87.0% on LIBERO-10, while remaining close on LIBERO-Object and LIBERO-Goal. The four-suite average rises from 94.15% to 94.35%. Notably, the strongest gain appears on LIBERO-10, which is consistent with the view that iterative decoding matters most on longer-horizon tasks.

C. Threshold scans reveal task-dependent non-monotonicity

Proposition 1 motivates residual thresholding because it bounds the same normalized residual r_k used in the experiments. Fig. 2 shows scans on `place_dual_shoes` and `click_bell`. For `place_dual_shoes`, success peaks near $\tau = 1.0$; for `click_bell`, the best result occurs at the strictest tested threshold. The preferred threshold is therefore task dependent.

Fig. 3 gives a single-task view on `click_alarmclock`. EqM inference passes through *early attraction*, *semantic shaping*, and *over-refinement*. We use *stationarity–*

TABLE I

DEPLOYMENT SUCCESS RATES ON THE 19-TASK ROBOTWIN BENCHMARK UNDER A MATCHED 300-STEP BUDGET. THE TWO METHODS SHARE THE SAME UPSTREAM VLA STACK; ONLY THE ACTION DECODER DIFFERS.

Task	π_0	π_0-EqM	Δ	Task	π_0	π_0-EqM	Δ
beat_block_hammer	88	85	−3	place_a2b_right	30	30	0
click_bell	38	33	−5	put_bottles_dustbin	42	72	+30
click_alarmclock	24	45	+21	stack_blocks_two	58	68	+10
handover_block	24	27	+3	stack_bowls_three	68	60	−8
move_can_pot	6	14	+8	turn_switch	34	34	0
move_playingcard_away	66	63	−3	pick_diverse_bottles	20	47	+27
place_phone_stand	48	55	+7	place_dual_shoes	54	45	−9
place_can_basket	36	50	+14	pick_dual_bottles	26	66	+40
place_cans_plasticbox	32	65	+33	place_object_stand	48	56	+8
place_a2b_left	26	39	+13				
Average	40.4	50.2	+9.8				

TABLE II

SUCCESS RATES (%) ON THE FOUR LIBERO BENCHMARK SUITES.

Suite	π_0	π_0-EqM	Δ
LIBERO-Spatial	96.8	97.2	+0.4
LIBERO-Object	98.8	98.4	−0.4
LIBERO-Goal	95.8	94.8	−1.0
LIBERO-10	85.2	87.0	+1.8
Average	94.15	94.35	+0.20

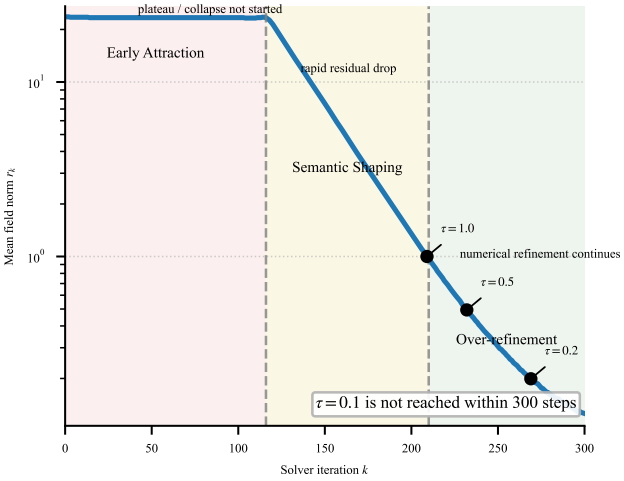

Fig. 3. Qualitative EqM inference trajectory on click_alarmclock, showing early attraction, semantic shaping, and over-refinement.

executability gap for the mismatch between numerical stationarity and physical utility: the iterate with the smallest residual need not execute best.

Together, Proposition 1 and the threshold scans suggest a two-part reading of early stopping: residual thresholds monitor solver progress, but the best threshold remains task dependent. In closed loop, the threshold is therefore not only a numerical tolerance; it also shapes the deployed action distribution.

VI. DISCUSSION AND LIMITATIONS

The experiments support three observations: EqM improves RoboTwin under matched compute, transfers competitively to LIBERO, and makes stopping depth behaviorally relevant once decoding becomes iterative. The main limitation is scope: the threshold scans cover only two RoboTwin tasks, so the non-monotonic phenomenon is illustrated rather than mapped exhaustively. Proposition 1 is likewise local: it justifies residual thresholding but does not identify the threshold that maximizes task success or provide a global convergence theorem for the learned non-conservative field. More practically, the current study also does not isolate how much of the gain comes from the EqM objective itself versus the stopping rule and warm-start structure.

At a systems level, the main implication is a cleaner separation between representation learning and deployment-time compute allocation. Once the decoder is time free, one can vary thresholds, iteration caps, or warm-start policies without changing the upstream VLA stack, which turns part of test-time compute into a controllable systems variable rather than a fixed sampler schedule. This perspective may be especially useful in future multi-task and cross-embodiment settings, where the same task-conditioned action field may need to be solved under different latency budgets, action granularities, or robot morphologies. These limitations and opportunities together reinforce the central point that compute allocation in closed-loop VLA should be evaluated behaviorally as well as numerically.

VII. CONCLUSION

We presented π_0-EqM, a decoder replacement for π_0 that casts action generation as iterative equilibrium solving. Under a matched 300-step budget, it improves RoboTwin average success from 40.4% to 50.2% across 19 tasks while remaining competitive on LIBERO and improving LIBERO-10 from 85.2% to 87.0%. More broadly, the results argue that inference depth is a policy variable in closed-loop VLA rather than a purely numerical afterthought. They also suggest energy-based VLA—especially composable action generation across tasks and embodiments—as a promising direction for future work.

REFERENCES

[1] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, *et al.*, "RT-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[2] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, *et al.*, "RT-2: Vision-language-action models transfer web knowledge to robotic control," in *Conference on Robot Learning*. PMLR, 2023, pp. 2165–2183.

[3] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, *et al.*, "Octo: An open-source generalist robot policy," *arXiv preprint arXiv:2405.12213*, 2024.

[4] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, *et al.*, "OpenVLA: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024.

[5] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, *et al.*, "π_0: A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024.

[6] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," *arXiv preprint arXiv:2210.02747*, 2022.

[7] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, vol. 44, no. 10-11, pp. 1684–1704, 2025.

[8] R. Wang and Y. Du, "Equilibrium matching: Generative modeling with implicit energy-based models," *arXiv preprint arXiv:2510.02300*, 2025.

[9] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," *arXiv preprint arXiv:2304.13705*, 2023.

[10] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1168–1175.

[11] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable mpc for end-to-end planning and control," *Advances in neural information processing systems*, vol. 31, 2018.

[12] Y. Du and I. Mordatch, "Implicit generation and modeling with energy based models," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[13] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson, "Implicit behavioral cloning," in *Conference on Robot Learning*. PMLR, 2022, pp. 158–168.

[14] T. Chen, Z. Chen, B. Chen, Z. Cai, Y. Liu, Z. Li, Q. Liang, X. Lin, Y. Ge, Z. Gu, *et al.*, "RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation," *arXiv preprint arXiv:2506.18088*, 2025.

[15] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone, "LIBERO: Benchmarking knowledge transfer for lifelong robot learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 44 776–44 791, 2023.